Titre: Title:	Confection d'horaires des médecins en salle d'urgence par une méthode hybride de génération de colonnes et de programmation par contraintes
Auteur: Author:	Hocine Lebbah
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lebbah, H. (2004). Confection d'horaires des médecins en salle d'urgence par une méthode hybride de génération de colonnes et de programmation par contraintes [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7191/

UNIVERSITÉ DE MONTRÉAL

CONFECTION D'HORAIRES DES MÉDECINS EN SALLE D'URGENCE PAR UNE MÉTHODE HYBRIDE DE GÉNÉRATION DE COLONNES ET DE PROGRAMMATION PAR CONTRAINTES

HOCINE LEBBAH
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2004

Your file Votre référence
ISBN: 0-612-91953-6
Our file Notre référence
ISBN: 0-612-91953-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONFECTION D'HORAIRES DES MÉDECINS EN SALLE D'URGENCE PAR UNE MÉTHODE HYBRIDE DE GÉNÉRATION DE COLONNES ET DE PROGRAMMATION PAR CONTRAINTES

présenté par: LEBBAH Hocine

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. GALINIER Philippe, Doct., président
M. PESANT Gilles, Ph.D., membre et directeur de recherche
M. GENDRON Bernard, Ph.D., membre et codirecteur de recherche
M. ROUSSEAU Louis-Martin, Ph.D., membre

à la mémoire de ceux qui nous sont chers

REMERCIEMENTS

Je tiens d'abord à remercier mes directeurs M. Gilles PESANT et M. Bernard GENDRON pour m'avoir soutenu sur tous les plans et avec confiance tout au long de la réalisation de ma maîtrise. Leurs idées complémentaires et leurs remarques constructives ont enrichi à la fois le contenu et la présentation de ce mémoire. Je les remercie pour m'avoir épaulé infatigablement lors de ma rédaction. Qu'ils trouvent ici l'expression de mon estime profonde et sincère.

Mes remerciemments vont aussi à M. Philippe GALINIER pour avoir accepté de présider mon jury. Je remercie également L. M. ROUSSEAU pour ses orientations et pour avoir accepté de juger mon travail.

Je tiens à remercier particulièrement Serge BISAILLON pour son aide et sa grande disponibilité. Je remercie également l'équipe des ressources informatiques du C.R.T pour leur support technique.

Je remercie aussi Stéphane NOLIN, Stéphane BOURDAIS et Marc BRISSON pour l'aide qu'ils m'ont apportée.

Que ma femme, qui n'a jamais cessé de me nourrir sur tous les plans, trouve ici l'expression de mes sentiments les plus respectueux.

Enfin, je remercie tous ceux et toutes celles qui ont contribué de près ou de loin à la réalisation de ce mémoire.

RÉSUMÉ

La confection d'horaires de médecins dans une salle d'urgence d'un grand hôpital constitue une tâche très complexe. Ces horaires doivent obéir à de nombreuses règles visant à maximiser la qualité des soins et les requêtes du personnel. Très souvent, les gestionnaires font face à la difficulté d'obtenir des horaires respectant toutes les contraintes étant donné le caractère conflictuel de celles-ci.

Ce mémoire de maîtrise porte sur la confection d'horaires de médecins dans une salle d'urgence. Nous proposons une approche basée sur une méthode hybride utilisant la génération de colonnes et la programmation par contraintes.

La résolution du modèle mathématique proposé présente deux difficultés principales qui s'ajoutent au caractère combinatoire du problème à l'étude:
la dégénérescence et l'intégralité. Ainsi, après un temps d'exécution significatif dans la phase d'optimisation, la valeur de la fonction objectif demeure inchangée. De plus, la recherche d'une solution entière échoue, fournissant souvent comme meilleure solution entière la solution initiale. Pour contrer ces problèmes, nous avons proposé des stratégies de recherche de solutions qui permettent une meilleure coopération entre le problème maître et le sous-problème. Nous avons proposé d'abord une stratégie basée sur les valeurs duales. Celle-ci vise essentiellement à faire évoluer la fonction objectif. En effet, dans la majorité des cas, le test est concluant, puisque nous avons observé dans les résultats présentés une amélioration

significative de la fonction objectif. À noter que cette stratégie est générale et peut être exploitée dans les cas où l'optimisation est capitale et le problème d'intégralité est secondaire. Même si cette stratégie permet de faire évoluer la fonction objectif dans la relaxation continue, le problème d'identifier de meilleures solutions entières demeure. La deuxième stratégie vise à aider le problème maître à trouver des solutions entières. Pour ce faire, nous avons orienté la recherche pour essayer de trouver des horaires tels que les contraintes du problème maître soient vérifiées et qu'en même temps la fonction objectif évolue. Nous acceptons alors d'augmenter le problème maître non seulement avec des colonnes intéressantes au sens des coûts réduits, mais aussi avec des colonnes permettant de favoriser l'obtention de solutions entières. Comme le montrent les résultats, avec cette dernière stratégie, la fonction objectif évolue et la solution entière obtenue est souvent différente de la solution initiale.

ABSTRACT

Building work schedules for emergency room physicians at a large hospital is a complex task. These schedules must respect many rules aiming at maximizing the quality of the health care provided but also the satisfaction of individual requests by members of the personnel. Very often, managers are faced with the impossible task of satisfying conflicting rules. This master's thesis addresses the problem of emergency room physician scheduling. We propose a hybrid approach combining column generation and constraint programming.

Solving the mathematical model proposed poses two main challenges, aside from its highly combinatorial nature: degeneracy and integrality. With a default search strategy, the objective value for a linear relaxation of the problem remains unchanged even after long computing times. No better integral solution is thus found. A major part of this work was to design search strategies that would increase the cooperation between the master problem and the subproblems. Our first strategy was based on dual values from the master problem in order to decrease the objective value. Such a strategy did improve the results of the relaxation in most cases but it remained difficult to produce better integral solutions. Our second strategy attempted to preserve the improvements on the results of the relaxation while generating solutions to the individual subproblems which combined well into an integral solution to the problem as a whole. Experiments confirm an

improvement of the integral solution.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

PC programmation par contraintes

CSP problème de satisfaction de contraintes ("Constraints Satisfaction Problem")

PLNE programme linéaire en nombres entiers

PL programme linéaire

LISTE DES TABLEAUX

INTRODUCTION

La confection d'horaires de médecins dans une salle d'urgence d'un grand hôpital constitue une tâche très complexe. Ces horaires doivent obéir à de nombreuses règles visant à maximiser la qualité des soins et les requêtes du personnel. Selon les établissements, l'objectif est une combinaison variable de considérations en termes de coûts, de qualité de soins et de satisfaction du personnel. Très souvent, les gestionnaires font face à la difficulté d'obtenir des horaires respectant toutes contraintes étant donné le caractère conflictuel de celles-ci. Notre travail reprend la situation décrite par Beaulieu[6] concernant la salle d'urgence de l'Hôpital Sacré-Coeur de Montréal. Nous proposons une approche différente basée sur sur une méthode hybride utilisant la génération de colonnes et la programmation par contraintes.

Dans le premier chapitre, nous présentons une vue générale des problèmes de confection d'horaires en milieu hospitalier. Nous donnons un aperçu des objectifs visés et les contraintes rencontrées. Nous terminons ce chapitre par une revue de littérature sur le domaine d'étude.

Le second chapitre est consacré à la présentation des concepts de base de la programmation par contraintes et de la technique de génération de colonnes. Nous introduisons comment décrire un problème de satisfaction de contraintes et les algorithmes de résolution de ce dernier. Nous présentons aussi les contraintes globales

utilisées dans le cadre de notre projet. Enfin, nous donnons le principe de fonctionnement, ainsi que l'algorithme de résolution de la technique de génération de colonnes.

Le troisième chapitre décrit le contexte de notre étude, soit la situation décrite par Beaulieu[6] concernant la salle d'urgence de l'Hôpital Sacré-Coeur de Montréal. Il introduit les termes et les notations utiles pour la compréhension des chapitres suivants. Il décrit, en particulier, les caractéristiques des tâches à effectuer et celles du personnel en place.

Le chapitre 4 concerne notre approche de résolution basée sur la technique de génération de colonnes. D'une part, après description des différentes variables, des contraintes et de la fonction objectif, nous présentons le modèle mathématique constituant le problème maître. D'autre part, nous présentons le modèle de programmation par contraintes qui a servi pour résoudre le sous-problème. Nous expliquons comment nous avons tenu compte des coûts réduits pour générer des colonnes potentielles. Enfin, nous terminons par la présentation des différentes stratégies de recherche utilisées pour résoudre le sous-problème.

Le chapitre 5 est consacré à l'implémentation et la présentation des résultats. Nous introduisons le langage utilisé, le fonctionnement général du système pour ensuite finir avec une analyse des résultats obtenus.

CHAPITRE 1

PROBLÈME DE CONFECTION D'HORAIRES DANS LE MILIEU HOSPITALIER

Dans ce chapitre, nous présentons le problème de confection d'horaires en mettant en exergue les principaux objectifs visés, ainsi que les contraintes dont il faut tenir compte. Nous finissons ensuite avec une revue de la littérature portant sur les problèmes d'horaires de personnel dans le domaine de la santé.

1.1 Présentation générale des problèmes de confection d'horaires

Étant donné un ensemble de tâches ou quarts de travail, la confection d'horaires consiste à affecter le personnel disponible de telle sorte que tous les quarts de travail soient assurés. La gestion de la confection d'horaires de travail est une occupation majeure à laquelle sont confrontées la plupart des organisations. Dans certains hôpitaux, celle-ci s'effectue aujourd'hui encore de façon manuelle. Pour ce faire, une personne experte est monopolisée pour effectuer la tâche. La prise en compte d'un grand nombre de contraintes, souvent conflictuelles, rend la tâche ardue. Dans un contexte de pénurie de personnel, le responsable fait face à la difficulté d'établir des horaires réalisables. Plusieurs facteurs contribuent à rendre la situation en milieu hospitalier encore plus complexe:

- Les soins doivent être assurés 7 jours sur 7 et 24 heures sur 24. Cette situation contraint le personnel à effectuer des tranches horaires considérées comme peu "désirables". Il y a lieu alors de répartir certains quarts de la manière la plus équitable possible.

- La demande de soins peut varier d'une période à une autre.

- L'affectation du personnel doit tenir compte des demandes individuelles du personnel, comme les demandes de congés, de jours de repos, etc...

- Le personnel est classé en catégories selon les compétences et les qualifications de tout un chacun. L'affectation du personnel doit se faire selon ces catégories.

- Les horaires déjà planifiés sont souvent perturbés par l'imprévisibilité des absences.

La capacité d'adaptation des ressources en personnel pour faire face à une demande changeante influe directement sur l'efficacité du système et la qualité du service offert. Généralement, dans le budget relié aux opérations d'un système de santé, la plus grande part est consacrée aux coûts reliés à l'emploi du personnel[36]. L'avènement des ordinateurs, avec leur grande puissance de calcul, doublé de méthodes de résolution de plus en plus raffinées et variées amènent les gestionnaires à s'orienter vers la génération automatisée des horaires. Le recours à ces moyens permet aux gestionnaires d'utiliser un nombre minimal et suffisant de personnel pour éviter des périodes de surplus inutiles ou des situations de manque de

personnel, et pour garantir la qualité des soins et la continuité des services, tout en respectant les contraintes organisationnelles et réglementaires.

Les nombreuses contraintes sont souvent conflictuelles: les satisfaire toutes devient une tâche impossible. Il est alors nécessaire de considérer le degré d'importance de chacune d'elles pour dégager celles que l'on pourrait violer, si nécessaire. Chaque milieu hospitalier, selon ses spécificités et ses objectifs, fixe les contraintes qu'il faut obligatoirement respecter et celles qu'il est possible de violer. Pour respecter du mieux qu'on peut une contrainte non obligatoire, celle-ci peut faire partie d'une fonction objectif. De cette façon, le degré de satisfaction de celle-ci sera le meilleur possible.

Il existe trois types d'horaires que l'on peut générer. On peut générer un horaire sur une période courte en se basant sur des patrons d'horaires, que les membres du personnel suivent à tour de rôle. Après une rotation complète, chacun des membres a fait les mêmes quarts et dans le même ordre. Un tel horaire est dit ***cyclique avec rotation***. Si l'horaire est généré pour une durée courte à partir de patrons d'horaires et que l'on considère que chaque membre répète son propre horaire, on parle alors d'horaire ***cyclique sans rotation***. On parle d'horaire ***non cyclique*** quand celui-ci est reconstruit de nouveau à chaque début de période de planification, qui est souvent plus longue que la période considérée lors de la planification d'un horaire cyclique. L'avantage d'un horaire non cyclique réside dans sa flexibilité permettant de tenir compte, entre autres, des préférences individuelles.

1.2 Revue de la littérature

La confection d'horaires a concerné en premier lieu le personnel infirmier. Plusieurs méthodes ont été appliquées: certaines sont basées sur l'utilisation d'un chiffrier électronique [1,43]; d'autres sur la programmation mathématique [2,11–13,27,28,33,35,45,48]. Quelques travaux ont utilisé la programmation par contraintes (PC) [9,16,22]. Enfin [32,42] ont proposé des méthodes hybrides combinant la recherche locale et la PC. Parmi les références les plus récentes, mentionnons l'étude effectuée par Berrada[7] pour la confection d'horaire basé sur trois relèves (jour, soir, nuit) sur une période de quinze jours. Pour tenir compte des congés fériés et des requêtes du personnel, habituellement difficiles à considérer, l'auteur a proposé deux modèles multi-objectifs en nombres entiers; le premier à multi-objectifs linéaires, est résolu par une méthode exacte. Le second à multi-objectifs non linéaires est traité par une heuristique de recherche tabou [21]. Nabli [34] se base sur le même modèle et propose une autre adaptation de recherche tabou en combinant les différentes approches de diversification et de selection du meilleur voisinage. Gagné [18] résout le même problème multi-objectifs par une méthode exacte en combinant la méthode de pondération de Sherali[40] à la méthode séquentielle, qui consiste à ordonner les objectifs par ordre d'importance. Les horaires générés sont de meilleure qualité que ceux générés par la méthode de Nabli[34] avec un temps de résolution semblable. Jaumard, Semet, Vovor [27] proposent un modèle de programmation linéaire généralisée et utilisent la technique de génération de colonnes

pour sa résolution. Le problème maître est un programme linéaire en nombres entiers ($PLNE$). Les colonnes susceptibles d'améliorer la fonction objectif sont générées en utilisant un algorithme de calcul du plus court chemin avec contraintes supplémentaires. Cette méthode est utilisée pour des unités de soins d'une cinquantaine d'infirmiers et sur un horizon de planification de six mois par tranche de quinze jours. Labit [28] a aussi utilisée une méthode basée sur la technique de génération de colonnes. Le sous problème est formulé comme un problème de plus court chemin dans un graphe acyclique avec contraintes sur les arcs. Meyer [32] a considére la confection d'horaire comme un probléme d'optimisation de contraintes hierrarchiques. Une hierrarchie de contraintes a été définie. Le niveau 0 constitue les contraintes rigides du problème qu'il est souhaitable de ne pas violer du tout. sa méthode, combinant la recherche locale et la PC, vise à essayer de trouver des horaires qui minimisent les violations des contraintes selon l'ordre d'importance des contraintes.

Au cours des dernières années, beaucoup de travaux basés sur la PC ont vu le jour. Heus[22] propose une approche de résolution basée sur les techniques de PC. Grâce à des options interactives, le système permet de tenir compte des préférences individuelles. La taille de l'effectif est de trente personnes et la période de planification est de quinze jours. Feuilletin [16] a repris le même problème que celui traité par Jaumard, Semet et Vovor[27]. Ce problème est modélisé sous forme de problème de satisfaction de contraintes (ou CSP, voir chapitre 2). Ce travail a

servi de base à Bourdais[9], qui a proposé une approche commune pour un ensemble d'hôpitaux de Montréal pour lesquels n'avaient pas été développé des méthodes spécifiques. Bourdais, Galinier et Pesant[10] ont proposé un ensemble de contraintes globales applicables sur un probème de confection d'horaire. L'approche utilisée combine la *PC* et les techniques de recherche locale.

Concernant la confection d'horaires pour les médecins dans une salle d'urgence, peu de travaux ont vu le jour jusqu'à présent. Pour résoudre ce problème, Vassilacopoulos[47] a proposé un modèle stochastique basé sur un système *M/G/m(t)*. L'étude de Lapierre et Carter[30] a porté sur six hôpitaux de la région de Montréal. Ils situent la planification d'horaires pour les médecins par rapport à celle du personnel infirmier. Dans le cas du personnel infirmier, en plus du respect de la convention collective, les horaires doivent minimiser la masse salariale et maximiser la satisfaction du personnel simultanément. Dans le cas des médecins, seule la satisfaction de l'effectif est importante. Beaulieu, Ferland, Gendron et Michelon[5] ont étudié le cas de l'hôpital Sacré-Coeur de Montréal. Ils ont proposé un modèle *PLNE* à plusieurs objectifs. L'horizon de planification considéré est de un mois. Le modèle obtenu en agrégeant les différents objectifs est résolu par une méthode exacte. Pour assurer la réalisabilité du modèle et essayer d'accroître le niveau de satisfaction des médecins, un ordre d'importance est défini au niveau des contraintes. Au besoin, une contrainte peut être retirée. Forget[17] reprend et généralise le même modèle. L'étude a porté sur les données de l'hôpital Sacré-

Coeur et celles de l'hôpital Santa-Cabrini. Saadie[38] propose un modèle CSP pour traiter le cas de l'hôpital Sacré-Coeur de Montréal et prend comme base le travail de Beaulieu[6]. Rousseau , Pesant et Gendreau ont proposé une approche hybride pour la confection d'horaires pour les médecins de l'Hôpital Santa-Cabrini. La méthode consiste à appliquer en premier un algorithme génétique pour trouver des solutions partielles pour ensuite appliquer le "Branch and Bound" de la PC. L'approche hybride utilisée a donné de meilleurs résultats que ceux obtenus avec l'une ou l'autre des méthodes combinées.

Notre étude est également basée sur le contexte décrit dans Beaulieu[6]. L'approche utilisée consiste à combiner la programmation mathématique et la PC.

CHAPITRE 2

MÉTHODOLOGIES

Ce chapitre présente, d'une part, les notions de base de la programmation par contraintes et, d'autre part, une brève description de la technique de génération de colonnes.

2.1 Programmation par contraintes

Au cours de ces dernières années, la PC, conçue pour résoudre des CSP, a démontré avec succès ses capacités à résoudre des problèmes combinatoires. La notion de contrainte apparaît dans plusieurs types de problèmes de la vie de tous les jours. Ce chapitre présente les notions de base de ce domaine[4, 8, 22, 31, 37, 41].

2.1.1 Variables et contraintes

Une contrainte est une relation logique entre différentes inconnues, appelées variables, chacune prenant ses valeurs dans un espace donné, appelé domaine. Ainsi, les valeurs que peuvent prendre simultanément les variables doivent respecter la propriété définie par la contrainte. Par exemple, la contrainte $x = y + 1$ restreint les valeurs que l'on peut affecter simultanément aux variables x et y de sorte que la valeur de x doit toujours dépasser de un la valeur de y. En particulier, si y prend la valeur 5, x est contraint d'avoir la valeur 6. Étant donné qu'une contrainte est

relationnelle, la connaissance de l'une des deux variables détermine l'autre. Ainsi, si x prend la valeur 8, y est contraint d'avoir la valeur 7.

Une contrainte peut être définie ***en intention***, à l'aide de propriétés mathématiques connues; par exemple : $x < y$ ou $A \vee B \Rightarrow \overline{C}$. Elle peut également être définie ***en extension***, en énumérant les tuples de valeurs appartenant à la relation. Par exemple, une certaine relation entre x et y peut être définie par

$$\{(0,1),(0,5),(2,3),(3,3),(3,5)\}.$$

Le nombre de variables composant une contrainte est appelé ***arité*** de cette dernière. Une contrainte est dite ***unaire*** ou d'***arité*** 1, si elle porte sur une seule variable, comme dans $x^2 - x = 1$. Elle est dite ***binaire*** ou d'***arité*** 2 si elle porte sur 2 variables, comme dans $x = y + 1$. Elle est ***ternaire*** si le nombre de variables est 3, comme dans $x + y = z$. Le nombre de variables composant une contrainte est appelé ***arité*** de cette dernière. La plupart des travaux existants portent sur des problèmes dont les variables varient dans des domaines finis.

2.1.2 Problèmes de satisfaction de contraintes

Un CSP est défini par la donnée d'un ensemble de variables, associé chacune à un domaine fini de valeurs et d'un ensemble de contraintes. Une contrainte est définie sur un sous-ensemble de variables et impose des restrictions sur la combinaison des valeurs que les variables concernées peuvent prendre.

Formellement, un CSP est défini par un triplet (X, D, C) où :

$X = \{x_1, \ldots, x_n\}$ ensemble de n variables du problème;

$D = \{D_1, \ldots, D_n\}$ ensemble de n domaines, chaque variable x_i prenant sa valeur dans D_i;

$C = \{C_1, \ldots, C_m\}$ ensemble de m contraintes.

Une solution de ce problème correspond à l'affectation d'une valeur à chaque variable prise dans son domaine de telle sorte que toutes les contraintes soient respectées.

2.1.3 Quelques exemples exprimés sous forme de contraintes

Exemple 1 (Coloriage de carte) Il s'agit de colorier les différents pays d'une carte géographique de sorte que si deux pays ont une frontière commune alors ils sont de couleurs différentes. On dispose pour cela des quatres couleurs suivantes : bleu, rouge, jaune et vert.

On associe une variable x_i différente par pays i à colorier, le domaine de chaque variable étant un ensemble de quatre couleurs:

$D(x_i) = \{bleu, rouge, vert, jaune\}$

Il y a une seule contrainte pour ce problème : deux régions voisines doivent être de couleurs différentes

$C = \{x_i \neq x_j \mid x_i$ et x_j sont deux variables de X correspondant à des pays voisins$\}$.

Exemple 2 (Problème des 8 reines)

L'énoncé en langage naturel est le suivant : trouver toutes les manières de placer 8 reines sur un échiquier 8 x 8 de telle sorte qu'elles ne soient pas mutuellement en prise (règle de prise classique des échecs).

On doit tout d'abord identifier l'ensemble des variables et leurs domaines de valeurs, et ensuite les contraintes qui relient ces variables. Une façon de faire est de prendre comme variable la position de chaque dame sur sa colonne. On a donc $X = \{x_1, \ldots, x_8\}$. Puisqu'il y a 8 positions possibles sur chacune des colonnes, on a donc $D_1 = \ldots = D_8 = \{1, 2, \ldots, 8\}$.

Pour être sûr que tout doublet de reines n'est pas sur la même ligne nous devons assurer que quels que soient i et j, l'expression $x_i \neq x_j$ est toujours vérifiée. Pour s'assurer que tout doublet de reines ne soit pas sur la même diagonale, nous devons garantir que quels que soient i et j, la condition $i - j \neq x_i - x_j$ et $i - j \neq x_j - x_i$ est toujours vérifiée.

Soit N la taille de l'échiquier, le CSP de ce problème est alors défini par :

$X = \{x_1, \ldots, x_N\}$

$D = \{D_1, \ldots, D_N\}$, avec $D_1 = \ldots = D_N = [1, N]$

$C = C_1 \cup C_2 \cup C_3$

$C_1 = \{x_i \neq x_j \quad | \quad (1 \leq i < j \leq N)\}$

$C_2 = \{x_i \neq x_j + (j - i) \quad | \quad ((1 \leq i < j \leq N))\}$

$C_3 = \{x_i \neq x_j - (j - i) \quad | \quad ((1 \leq i < j \leq N))\}$

2.1.4 Propagation de contraintes

La propagation de contraintes consiste à trouver un problème équivalent au problème de départ de sorte que la recherche de solutions se fasse sur un espace moins volumineux en éliminant des domaines de variables les valeurs ne conduisant à aucune solution.

Pour retirer des domaines des variables les valeurs incohérentes, plusieurs techniques, dites de *cohérences*, peuvent être utilisées.

La cohérence de noeud est appliquée dans le cas de contraintes unaires. Une variable x satisfait la cohérence de noeud si et seulement si, quelle que soit la valeur du domaine courant de cette variable, chacune des contraintes unaires liées à x est satisfaite.

La cohérence d'arc est utilisée dans le cas de contraintes binaires. Une contrainte binaire impliquant deux variables x et y satisfait la cohérence d'arc si et seulement si, pour toute valeur de l'une des deux variables, il existe au moins une valeur de l'autre variable qui satisfait la contrainte en question. De façon plus générale, dans le cas d'une contrainte arithmétique faisant intervenir k variables, on applique la cohérence d'arcs généralisée. Il s'agit de vérifier que pour toute valeur de n'importe quelle variable, il existe au moins une valeur pour chacune des $k-1$ autres variables telle que la contrainte soit vérifiée.

La cohérence de bornes est appliquée pour resserrer, si possible, les valeurs extrêmes des domaines des variables impliquées. Une contrainte satisfait la cohérence

de bornes, si pour chaque borne du domaine courant de chaque variable, il existe au moins une valeur pour chacune des autres variables telle que cette contrainte soit respectée.

Illustrons ces techniques de cohérence sur un exemple particulier simple. Soient deux variables x et y liées par la contrainte C_1 telle que $x + y = 2$. On suppose que $D_x = \{-4, -3, 0, \ldots, 10\}$ et $D_y = \{0, 1, \ldots, 15\}$. En appliquant la cohérence de bornes, les domaines de x et de y deviennent $D_x = \{-4, -3, 0, \ldots, 2\}$ et $D_y = \{0, 1, \ldots, 6\}$. En effet, si on considère en particulier la valeur 10 de x, il n'existe aucune valeur de y qui laisse la contrainte satisfaite. Donc, la borne 10 sera éliminée du domaine D_x. Ce constat est vrai pour les valeurs 9, 8, ..., 3 de x qui seront donc éliminées aussi. Pour la borne inférieure -4 de x, le problème ne se pose pas puisqu'il existe une valeur de y telle que c_1 soit vérifiée. Le même procédé est appliqué pour réduire le domaine de la variable y. À remarquer que la cohérence de bornes se limite uniquement à réduire, quand cela est possible, les valeurs extrêmes des domaines des variables. En ajoutant deux contraintes unaires C_2 telle que $x \neq 0$ et C_3 telle que $y \neq 5$, l'application de la cohérence de noeuds réduira le domaine de x en éliminant la valeur 0 et le domaine de y en éliminant la valeur 5. Les domaines deviennent alors $D_x = \{-4, -3, 1, 2\}$ et $D_y = \{0, 1, \ldots, 4, 6\}$. Enfin, en appliquant la cohérence d'arc, toutes les valeurs de x(respectivement de y) pour lesquelles il n'existe pas de valeur de y (respectivement de x) telle que C_1 soit vérifiée seront retirées de D_x (respectivement de D_y). Dans notre cas, pour la

variable x, il n'y a que la valeur -3 de x qui sera éliminée puisqu'il n'existe pas de valeur de y telle que C_1 soit satisfaite. Par contre, pour la variable y, les valeurs 2, 3, 4 seront éliminées puisqu'il n'existe pas, pour aucune d'elles, de valeur de x vérifiant C_1. Les différentes techniques de cohérences appliquées ont permis de réduire les domaines des variables de x et de y à: $D_x = \{-4, 1, 2\}$ et $D_y = \{0, 1, 6\}$.

Dans certains cas, l'utilisation de certaines techniques de filtrages plus complexes peuvent être nécessaires. Il existe plusieurs techniques de cohérence d'arc qui peuvent être utilisées: cela va de l'AC-1 à l'AC-7. La différence entre elles réside dans leurs performances et leurs complexités. Les plus utilisées sont l'AC-3 et l'AC-4. Pour plus de détails sur ces techniques voir [4,8].

Deux points sont à considérer dans la conception d'algorithmes de propagation:

- l'*incrémentialité*, pour ne pas repartir à zéro à chaque fois qu'une contrainte est examinée de nouveau;
- l'équilibre entre le rapport quantité/fréquence de filtrage et le temps de calcul.

La propagation peut se faire suite à une modification du domaine d'une variable (événement "Domain"), ou lorsque la valeur minimale ou maximale du domaine a changé (événement "Range") ou encore lorsque le domaine est réduit à une seule valeur (événement "Value").

2.1.5 Résolution d'un problème de satisfaction de contraintes

Les CSP se rattachent à la grande classe des problèmes d'optimisation combinatoire. Le nombre de combinaisons qu'il faut envisager pour trouver une solution qui respecte toutes les contraintes est très grand et parfois même infini. La grande puissance des ordinateurs ne suffit pas pour examiner toutes les combinaisons possibles en un temps raisonnable. Selon le besoin, nous pouvons chercher soit une solution du problème, soit toutes les solutions ou encore une solution optimale par rapport à un objectif donné.

2.1.5.1 L'algorithme "Générer et Tester"

Cet algorithme consiste à générer l'ensemble des affectations complètes possibles et ensuite tester si elles satisfont les contraintes d'où l'appellation "Générer et Tester" . L'ensemble des affectations complètes est appelé l'espace de recherche du CSP. Si le domaine de certaines variables contient une infinité de valeurs, alors cet espace de recherche est infini et on ne pourra pas énumérer ses éléments en un temps fini. Néanmoins, même en se limitant à des domaines comportant un nombre fini de valeurs, pour des problèmes de grande taille, l'espace de recherche est souvent de taille tellement importante que cette méthode ne pourra se terminer en un temps raisonnable. Ainsi, le nombre d'affectations à générer croît de façon exponentielle en fonction du nombre de variables du problème. Les algorithmes suivants viseront à réduire la taille de l'espace de recherche.

2.1.5.2 L'algorithme "simple retour-arrière"

Une première façon d'améliorer l'algorithme "Générer et Tester" consiste à vérifier au fur et à mesure de la construction de l'affectation partielle sa consistance (i.e, vérifier si toutes les contraintes sont satisfaites) : dès lors qu'une affectation partielle est inconsistante, il est inutile de chercher à la compléter. Dans ce cas, on "retourne en arrière" ("backtrack" en anglais) jusqu'à la plus récente instanciation partielle consistante que l'on peut étendre en affectant une autre valeur à la dernière variable affectée. On itère le processus jusqu'à obtention de la solution complète. Cette méthode permet de réduire l'espace de recherche et permet donc un gain en temps. Cependant, sa rapidité de recherche reste insuffisante.

2.1.5.3 Méthodes rétrospectives "look back"

Alors que le retour-arriere revient sur l'instanciation de la variable précédent immédiatement la variable courante en cas de retour-arrière, les méthodes rétrospectives tentent d'identifier les causes de l'échec. Elles tirent parti de cette information soit pour sélectionner un meilleur point de retour ("backjumping", "graph-based backjumping", "dynamic backtracking", "conflict-directed backjumping"), soit pour identifier des instanciations partielles ne participant à aucune solution et ainsi éviter de refaire certaines affectations menant à un échec ("backmarking", "conflict-set", "back-checking").

2.1.5.4 Méthodes prospectives "look ahead"

Ces méthodes tirent pleinement parti des filtrages, c'est à dire élimination de valeurs des domaines de variables non liées, en les utilisant tout au long de la recherche. Leur principe est d'examiner les valeurs des variables non instanciées afin d'éliminer celles qui ne peuvent pas participer à un prolongement de l'instanciation partielle courante en une solution. Pour cela, on utilise une propriété de consistance locale et on retire des valeurs qui, compte tenu de l'instanciation courante, ne vérifient pas cette propriété. On élague ainsi l'arbre de recherche en supprimant des branches ne comportant pas de solutions. La méthode prospective la plus courante est le "forward-checking".

2.1.6 Heuristiques de recherche

Lors de l'utilisation d'un algorithme de recherche basé sur le retour-arrière, un ordonnancement efficace des variables et un bon choix de valeurs pour celles-ci peut avoir un impact positif important dans la résolution d'un CSP.

2.1.6.1 Choix de variables

L'ordre dans lequel on visite les nœuds, représentant les variables, de l'arbre de recherche est important et détermine le temps requis pour trouver une solution valide. L'ordre peut être statique si celui-ci est établi avant la recherche. Pour tirer profit des informations courantes de l'état d'un CSP, un ordre dynamique,

déterminé pendant la recherche, est souvent préféré.

Plusieurs heuristiques ont été développées pour établir cet ordre. Les plus communément utilisées sont basées sur le principe de l'échec d'abord "first-fail". Dans ce cas, l'ordre est établi dynamiquement en choisissant d'abord les variables où on a le maximum de chances d'échouer. En agissant de la sorte, l'incohérence est rapidement détectée. Une manière de faire est de choisir d'abord les variables dont le domaine est le plus petit. L'ordre peut être fait aussi en choisissant d'abord les variables participant au plus grand nombre de contraintes. Dans le cas de l'optimisation, une heuristique basée sur le principe du moindre regret est utilisée. Celle-ci consiste à choisir d'abord la variable dont la différence de coût entre les deux meilleures valeurs de son domaine est la plus grande. Étant donné ce choix où l'on a fait le mieux pour notre objectif, cette manière de faire permettra de minimiser les regrets. Après avoir choisi une variable, le choix de valeurs est l'étape importante qui suit.

2.1.6.2 Choix de valeurs

Après avoir choisi une variable, l'ordre dans lequel les valeurs de cette variable sont considérées peut avoir une influence importante sur le temps de résolution. En effet, pour un CSP ayant une solution unique, celle-ci peut être obtenue sans retour-arrière grâce à un choix de valeur bien guidé. Le principe du choix de valeur est à l'opposé du choix de variable. Il ne s'agit plus d'essayer d'échouer d'abord

mais d'essayer de réussir en premier lieu. Ainsi, la valeur qui a le plus de chance de réussir est choisie d'abord. Pour ce faire, il est important de tenir compte des spécificités du problème étudié pour trouver l'heuristique la plus adéquate. Il faut noter que l'ordre du choix de valeur influence l'ordre dans lequel les branches de l'arbre sont explorées. Donc, dans un contexte d'optimisation, si on est capable d'éliminer des branches entières en évaluant le coût à chaque point de choix, alors il peut être intéressant d'obtenir une bonne solution rapidement et ainsi couper les branches le plus tôt possible.

2.1.7 Contraintes globales

On appelle ***contrainte globale*** toute contrainte d'arité élevée pour laquelle il existe un algorithme de filtrage performant(de complexité polynomiale) qui maintient un haut niveau de cohérence entre les variables. Plusieurs contraintes globales ont été définies. Nous pouvons citer la contrainte $AllDiff(E)$ qui contraint toutes les variables d'un tableau E à prendre des valeurs différentes. Les contraintes globales sont des techniques qui permettent de traiter un sous-ensemble de contraintes du même type plus efficacement que les méthodes traditionnelles. Voici certaines contraintes globales qui ont servi pour l'écriture des contraintes dans ce projet.

Soient un vecteur $X = (x_1, x_2, ..., x_n)$ de variables à domaine fini et une variable Y supplémentaire à valeurs dans l'ensemble des entiers relatifs. La contrainte

sum $< y, X >$ assure que y est la somme des variables de X en appliquant la cohérence de bornes à la relation : $y = \sum_{i=1}^{n} x_i$, cette contrainte est d'une complexité linéaire.

Soient $X = (x_1, x_2, ..., x_n)$ de n variables à domaine fini dont les composantes prennent leurs valeurs dans l'ensemble $V = \{v_1, v_2, ..., v_m\}$ et un autre vecteur $C = (c_1, c_2, ..., c_m)$ de variables de comptage à domaine fini prenant leurs valeurs dans l'ensemble des entiers naturels. La contrainte **distribute**(C, V, X) assure que la distribution des valeurs de V soit telle que chaque v_j apparaisse c_j fois dans X. Cette contrainte maintient la cohérence d'arc généralisée et sa complexité est polynomiale[41]. À noter que la dimension de C doit être égale à celle de V. Aussi, si $V = \{1, 2, ..., m\}$ alors l'écriture devient **distribute**(C, X).

Soit $(s_1, s_2, ..., s_n)$ un vecteur d'éléments de V. On appelle *séquence* (maximale) de type $v \in V$ toute suite d'éléments consécutifs $s_i, s_{i+1}, ..., s_j$ telle que:

$$\begin{cases} s_i = s_{i+1} = ... = s_j = v \\ i > 1 \Rightarrow s_{i-1} \neq v \\ j < n \Rightarrow s_{j+1} \neq v \end{cases}$$

La taille d'une telle séquence est $j - i + 1$.

Soient $X = (x_1, x_2, ..., x_n)$ de n variables à domaine fini dont les composantes prennent leurs valeurs dans l'ensemble $V = \{v_1, v_2, ..., v_m\}$. Soient $\underline{\lambda}$ et $\overline{\lambda}$ deux vecteurs d'entiers de taille m. La contrainte **stretch**$(X, V, \underline{\lambda}, \overline{\lambda})$ assure que, dans

toute assignation valide de X, la taille des séquences de type $v_k, 1 \leq k \leq m$, est comprise entre $\underline{\lambda}_k$ et $\overline{\lambda}_k$. Lorsque $V = \{1, 2, ..., m\}$ alors l'écriture devient **stretch**$(X, \underline{\lambda}, \overline{\lambda})$. Cette contrainte possède une complexité polynomiale[37].

Enfin, la contrainte **ext**(X, Γ) définit l'ensemble Γ des tuples admissibles parmi tous les tuples de valeurs possibles pour le tuple de variables $(x_1, x_2, ..., x_n)$. Elle peut exprimer toute relation entre les variables de X. Cette contrainte maintient la cohérence d'arc généralisée, mais sa complexité est exponentielle. Cette contrainte est utilisée dans le cas où le nombre de variables est restreint.

2.2 Génération de colonnes

Dans cette section, nous donnons quelques notions de programmation linéaire et une brève présentation de la technique de génération de colonnes.

2.2.1 Notions sur la programmation linéaire

Un programme linéaire se présente sous la forme suivante:

$$\min_x c^T x \tag{2.1}$$

$$s.c. \quad Ax = b \tag{2.2}$$

$$x \geq 0 \tag{2.3}$$

x: est le vecteur colonne, de dimension n, représentant les variables de décision.

c^T: est le vecteur ligne représentant les coûts associés aux variables de décision.

A: est la matrice des coefficients du système d'équation représentant m contraintes.

b: est le vecteur colonne, de dimension m, représentant les valeurs des membres de droite des équations correspondant aux contraintes.

L'ensemble des solutions réalisables

$$S = \{x \in \Re^n : Ax = b, x \geq 0\} \tag{2.4}$$

forme un polyèdre. Un programme linéaire est réalisable si l'ensemble S est non vide. De plus, $x \in S$ est un point extrême s'il ne peut s'écrire comme une combinaison convexe stricte de deux points $x_1, x_2 \in S, x_1 \neq x_2$.

Définissons une matrice de base B comme étant toute sous-matrice de A formée de m colonnes linéairement indépendantes de A, et une matrice hors base N telle que $A = \{B|N\}$. On peut alors définir une solution de base associée à B comme étant le point $x_B \in \Re^n$ tel que:

$$x_B = B^{-1}b \tag{2.5}$$

et si $x_B \geq 0$, alors la solution est dite réalisable et elle correspond à un point extrême du polyèdre. Pour résoudre un programme linéaire, l'algorithme du simplexe est généralement utilisé. Celui-ci est un processus itératif qui consiste à itérer d'un

point extrême à un autre du polyèdre. À chaque itération, on cherche à améliorer la solution x_B par rapport à la fonction objectif. En somme, on cherche le point extrême du polyèdre qui correspond à la meilleure valeur de la fonction objectif. Quand le nombre de variables est beaucoup plus grand que le nombre de contraintes, une version révisée du simplexe est utilisée. Celle-ci permet un gain de temps considérable en ne calculant à chaque itération que la matrice de base courante. Le programme linéaire que nous avons à résoudre présente un nombre de variables très supérieur au nombre de contraintes. De plus, la matrice N n'est pas connue. Nous avons alors utilisé la technique de génération de colonnes qui est basée sur la version révisée de l'algorithme du simplexe.

2.2.2 Principe de fonctionnement de la méthode

L'idée de la génération de colonnes consiste à décomposer un problème linéaire de telle manière à le résoudre sans pour autant exprimer toutes les variables (colonnes). On applique alors la technique de décomposition de Dantzig-Wolfe [14], qui permet d'obtenir deux problèmes qui coopèrent jusqu'à l'obtention d'une solution optimale. La procédure consiste à démarrer avec un ensemble de variables de base d'un programme linéaire équivalent au problème initial. Ce programme est appelé *problème maître*. D'autre part, on résout un autre problème qui permet de trouver, si possible, une variable ou colonne du problème maître qui améliore la solution courante obtenue avec les colonnes déjà ajoutées. Ce problème est

dit *problème auxiliaire* ou *sous-problème.* Les variables ou colonnes générées sont celles dont le *coût réduit* est négatif (cas de minimisation). Afin d'évaluer les coûts réduits des colonnes non encore générées, la résolution du problème maître fournit au sous-problème les valeurs duales associées aux contraintes. la figure 2.1 illustre le principe de fonctionnement de la technique de génération de colonnes.

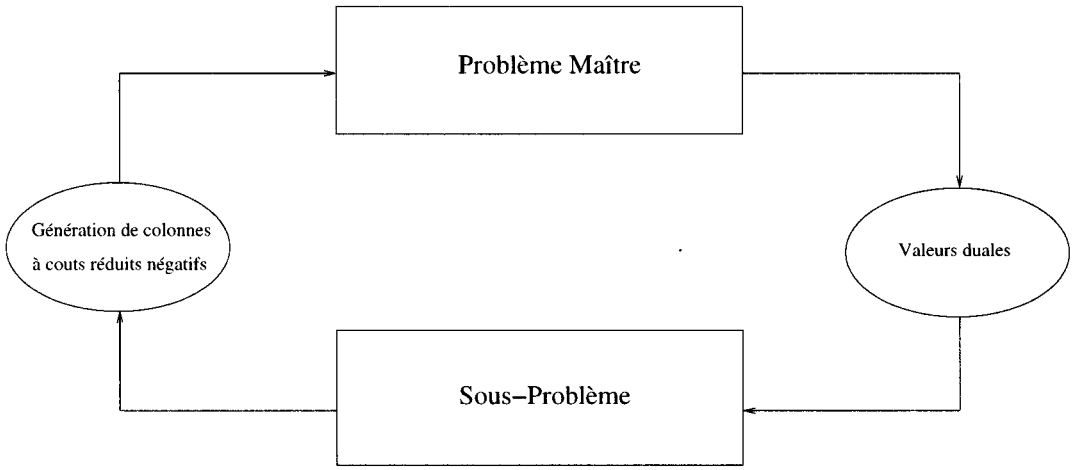

Figure 2.1: Une itération de la technique de génération de colonnes

2.2.3 Algorithme de la méthode

La méthode de génération de colonnes est appliquée lorsque le nombre de variables est beaucoup plus grand que le nombre de contraintes et donc très difficile à énumérer. Son but est de réduire la taille du problème à résoudre. Elle se base sur la méthode révisée du simplexe. L'algorithme de cette technique peut être présenté comme suit:

1. **Initialisation**: choisir une base $B = B^0$; itération $k = 0$;

2. $k \leftarrow k + 1$.

3. **Résoudre le problème maître restreint**:

   Calculer la solution de base courante $x_B = B^{-1}b$ et

   le vecteur des multiplicateurs du simplexe ou valeurs duales (π).

4. **Résoudre le sous-problème**:

   Trouver une nouvelle colonne (A_s) telle que $\overline{c}_s = c_s - \pi A_s$ et $\overline{c}_s < 0$ correspondant au coût réduit négatif;

5. Si $\overline{c}_s \geq 0$ alors terminer: la solution de base courante obtenue est la solution optimale; sinon, ajouter la nouvelle colonne A_s à B et recommencer avec la nouvelle base à partir de l'étape 2.

Dans l'algorithme du simplexe, le choix de la colonne de plus petit coût réduit se fait en énumérant toutes les variables hors base. Dans le cas où l'on ne connaît pas toutes les colonnes, la façon d'obtenir la nouvelle colonne est plus complexe et se fait par la résolution du sous-problème. Les méthodes utilisées varient. Souvent, on cherche à obtenir la colonne correspondant au coût réduit minimal en résolvant un problème de plus court chemin sous contraintes additionnelles à l'aide de la programmation dynamique. Dans notre cas, nous avons utilisé la programmation par contraintes.

Dans le cadre d'un problème en nombres entiers, la solution continue obtenue à l'issue de l'algorithme de génération de colonnes ne peut être qu'une borne

inférieure de la solution entière recherchée. Celle-ci pourrait être utilisée comme évaluation par défaut dans un algorithme de "Branch-and-Bound" qui cherche la solution optimale entière.

CHAPITRE 3

CONTEXTE D'ÉTUDE

Dans ce chapitre, nous présentons la problématique et nous donnons quelques définitions de termes, concernant les quarts de travail et les médecins, utiles pour la compréhension de la suite du mémoire.

3.1 Problématique

Nous nous intéressons en particulier à la confection d'horaires de médecins dans la salle d'urgence de l'Hôpital Sacré-Coeur de Montréal. Nous considérons le contexte de 1998 décrit par [6] où l'on dispose d'un effectif de dix-huit médecins pour assurer la prise en charge de chaque quart de travail pour chacun des jours de la période de planification. Pour confectionner cet horaire, nous devons tenir compte d'un certain nombre de contraintes. Celles-ci peuvent être réparties en deux catégories: celles qui sont obligatoires et celles qu'il est souhaitable de respecter, que nous pouvons considérer comme souples. Il est, par exemple, impératif que chaque quart soit pourvu par un et un seul médecin et ce, pour chaque jour de l'horizon de planification. Pour que les tâches soient accomplies avec efficacité, il faut éviter, pour tout médecin, les séquences trop longues de travail consécutif. Une nuit de travail ne doit jamais être suivie d'un travail le jour ou le soir du lendemain.

Certains quarts de travail sont plus exigeants que d'autres; alors, il est souhaitable de respecter un certain équilibre en tenant compte du degré de difficulté. La charge de travail est à répartir de la manière la plus équitable possible en tenant compte de l'ancienneté, des congés et des vacances de chacun des médecins. Par ailleurs, le planificateur doit tenir compte, dans la mesure du possible, des souhaits du personnel. Ceci permettrait à certains d'intervenir dans d'autres cliniques et à d'autres d'assurer des activités en matinée ou en soirée.

L'idéal est de trouver un horaire respectant parfaitement toutes les contraintes, mais une telle solution n'existe pas en général, étant donné que les contraintes sont souvent conflictuelles. Nous cherchons alors une solution de compromis visant à garantir le respect des contraintes dures ou obligatoires, et la minimisation des violations de chacune des contraintes souples.

3.2 Définitions des termes

Nous définissons ici les termes les plus usités pour la confection d'horaire.

3.2.1 Quart de travail

Un quart de travail est une tâche d'une durée bien déterminée. Il est défini par une heure de début et une heure de fin. On distingue deux types de quarts selon l'intensité du travail à effectuer. On parle de ***quart de choc*** quand la tâche à effectuer consiste à prendre en charge des patients présentant une urgence majeure.

Dans les autres cas, on parle de ***quart de cube*** (par exemple, un suivi de patients dans les chambres). On considère qu'un quart de choc est plus pénible qu'un quart de cube. Un quart de travail peut être fait de jour, de soir ou de nuit selon son heure de début et son heure de fin. Il est dit ***quart de jour***, si l'heure de début et l'heure de fin sont comprises entre 8:00 et 18:00 inclusivement. Il est dit ***quart de soir***, si l'heure de début et l'heure de fin sont comprises entre 16:00 et 24:00 inclusivement, et enfin, il est dit ***quart de nuit***, si l'heure de début et l'heure de fin sont comprises entre 24:00 et 8:00 inclusivement. Un quart est ***ouvert*** si, après planification, celui-ci demeure non assuré. Il est dit ***fixé*** si celui-ci est attribué à un médecin avant la planification, et il est dit ***variable*** s'il est affecté lors du processus de confection d'horaires. Le contexte de l'Hôpital Sacré-Coeur comprend huit quarts différents:

"8": Quart de choc et de jour se déroulant chaque jour de la semaine entre 8:00 et 16:00.

"10": Quart de cube et de jour se déroulant chaque jour de la semaine entre 10:00 et 18:00.

"16": Quart de choc et de soir se déroulant chaque jour de la semaine entre 16:00 et 24:00.

"16T": Quart de cube et de soir se déroulant chaque jour de la semaine entre 16:00 et 24:00.

"N": Quart de nuit se déroulant chaque jour de la semaine entre 24:00 et 8:00.

"8T": Quart de cube et de jour se déroulant chaque jour de la semaine entre 8:00 et 16:00.

"SS": Quart de cube et de jour se déroulant du lundi au vendredi entre 8:00 et 16:00.

"18": Quart de cube et de soir se déroulant chaque jour de la semaine entre 18:00 et 22:00 quand un besoin de désengorger la salle d'urgence est prévu.

Le tableau 3.1 illustre une semaine complète de travail représentant tous les quarts à couvrir.

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
"8"	"8"	"8"	"8"	"8"	"8"	"8"
"10"	"10"	"10"	"10"	"10"	"10"	"10"
"16"	"16"	"16"	"16"	"16"	"16"	"16"
"16T"	"16T"	"16T"	"16T"	"16T"	"16T"	"16T"
"N"	"N"	"N"	"N"	"N"	"N"	"N"
"8T"	"8T"	"8T"	"8T"	"8T"		
"SS"		"SS"		"SS"		
"18"	"18"	"18"	"18"	"18"	"18"	"18"

Tableau 3.1: Semaine avec présence de tous les quarts

3.2.2 Périodes

Nous distinguons différentes périodes à considérer lors de la confection de l'horaire. Ainsi, un ***jour*** est une période d'une durée de 24 heures qui s'étale de 8:00 à 8:00 le

lendemain. Une ***semaine*** est une période de sept jours commençant le lundi et se terminant le dimanche. Une ***fin de semaine*** est la période entre vendredi 16:00 et le lundi 8:00. La fin de semaine est modifiée si le vendredi, ou le lundi, est férié. La ***période de planification*** est une période de vingt-huit jours commençant lundi et finissant dimanche.

3.2.3 Activités reliées au travail

Une ***réunion***, durant laquelle les médecins abordent les différents aspects concernant la salle d'urgence et sa prise en charge, se tient une fois par semaine et a lieu, généralement, le jeudi matin. À chaque semaine, un ***cours*** est donné par le médecin affecté au quart "16T". Ce cours a lieu le mercredi soir. Un ***événement spécial***, liée au domaine médical et ayant pour conséquence de réduire l'effectif de médecins pour une période donnée, peut survenir. Il s'agit le plus souvent de colloques auxquels prennent part certains médecins.

3.2.4 Statut des médecins

Un médecin est considéré ***régulier*** s'il peut être soumis à toutes les règles de distribution des quarts. Son volume horaire hebdomadaire est plus ou moins fixé près d'un certain seuil. Un médecin est dit à ***temps partiel*** s'il n'a que quelques disponibilités pendant la période de planification. Sa charge hebdomadaire est souvent très légère et ses disponibilités sont prédéfinies et servent le plus souvent

à combler le manque de personnel.

Un médecin est considéré ***sénior*** à partir du moment où il a totalisé trois années de service en salle d'urgence. Autrement, un médecin est considéré ***junior***.

Pour prendre en charge la salle d'urgence, l'Hôpital Sacré-Coeur de Montréal dispose d'un total de dix-huit médecins dont seize sont réguliers et deux à temps partiel. Il y a cinq médecins juniors parmi les réguliers. Ainsi, la salle d'urgence fonctionne avec ce qui suit:

$I = \{0, 1, ..., 17\}$: ensemble de tous les médecins;

$I_{reg} = \{0, 1, ..., 15\}$: ensemble des médecins réguliers;

$I_{par} = \{16, 17\}$: ensemble des médecins à temps partiel;

$I_{jun} = \{0, 1, ..., 4\}$: ensemble des médecins juniors;

$I_{sen} = \{5, 6, ..., 15\}$: ensemble des médecins séniors.

Dans ce chapitre, nous avons présenté le contexte de notre étude. Le chapitre suivant sera consacré à la présentation de l'approche de résolution du problème posé.

CHAPITRE 4

APPROCHE DE RÉSOLUTION

Dans ce chapitre, nous présentons l'approche adoptée pour résoudre le problème de confection d'horaire à l'étude. Notre démarche consiste à utiliser une méthode hybride où les forces de la programmation par contraintes et celles de la programmation mathématique sont conjuguées pour venir à bout de l'objectif. Étant donné le très grand nombre de variables de notre formulation de programmation mathématique, nous avons opté pour la méthode de génération de colonnes qui nous permet de ne considérer que les colonnes intéressantes limitant ainsi le nombre de variables.

Nous comptons ainsi exploiter la force et la facilité d'expression de la programmation par contraintes pour tenir compte du grand nombre de contraintes, souvent difficiles à exprimer, et des spécificités de chacun des médecins. Nous utilisons la programmation mathématique pour produire des horaires de bonne qualité, qui minimisent, au mieux possible, les violations des contraintes souples et qui garantissent en même temps la satisfaction de la demande. Les deux approches fonctionnent de manière interdépendante. L'idéal est de trouver un horaire respectant parfaitement toutes les contraintes, mais une telle solution n'existe pas, étant donné que les contraintes sont souvent conflictuelles. Nous chercherons alors une solution de

compromis visant à garantir le respect des contraintes dures ou obligatoires et la minimisation des violations de chacune des contraintes souples.

4.1 Problème maître

Dans cette section, nous allons définir les variables de décision et les contraintes, ainsi que la fonction objectif composant le problème maître.

4.1.1 Variables

Le processus de génération de colonnes consiste à générer plusieurs horaires individuels pour chacun des médecins pour ensuite en choisir un seul par médecin de telle sorte que la contrainte de quota soit satisfaite. Ainsi, il y a lieu de décider lequel fera partie de la solution, sachant qu'un horaire est soit choisi complètement, ou pas du tout. Nos variables de décision sont alors définies comme suit:

$$x_i^h = \begin{cases} 1, & \text{si l'horaire } h \text{ est affecté au médecin } i, \\ 0, & \text{sinon,} \end{cases}$$

$\forall\, i \in I$, l'ensemble des médecins, et

$\forall\, h \in H_i$, l'ensemble des horaires possibles pour le médecin i.

4.1.2 Contraintes

En premier lieu, nous devons nous assurer que chaque quart de l'horizon de planification est assuré une et une seule fois. La contrainte correspondante s'exprime alors de la façon suivante:

$\sum_{i \in I} \sum_{h \in H_i} \delta_h^k x_i^h = 1, \quad \forall\, k \in K,$ l'ensemble des quarts de travail,

où

$$\delta_h^k = \begin{cases} 1, & \text{si le quart } k \text{ appartient à l'horaire } h, \\ 0, & \text{sinon.} \end{cases}$$

Par ailleurs, nous devons garantir que le planning généré est composé d'un et un seul horaire par médecin. Cette contrainte est exprimée comme suit:

$\sum_{h \in H_i} x_i^h = 1, \quad \forall\, i \in I.$

4.1.3 Fonction de coût

Dans cette section, nous présentons en premier lieu les différentes contraintes de buts que nous avons considérées, et ensuite, nous donnons l'expression de la fonction de coût.

Les contraintes de buts que nous avons considérées constituent une partie de l'ensemble des contraintes présentées par Beaulieu[6]. Pour chacune des contraintes, on définit une valeur souhaitée ou but. L'idéal est que la valeur effective correspondant à un horaire donné soit égale à la valeur souhaitée et dans ce cas,

la pénalité est nulle. Nous avons défini deux poids pour chacune des contraintes, selon que la différence entre la valeur effective et la valeur souhaitée est positive ou négative. Plus la valeur effective s'éloigne de la valeur souhaitée, plus la pénalité devient importante. La pénalité relativement à une contrainte de but donnée est déterminée par le produit de la différence positive, entre la valeur effective et la valeur souhaitée, par le poids associé. Les contraintes de buts considérées sont les suivantes:

- **Rapport cube/choc.** Vu la différence de difficulté entre un quart de choc et un quart de cube, il est souhaité que chaque médecin, lors de la période de planification, assure un nombre de quarts de choc et de cube tel que le rapport entre le nombre de quarts de choc effectués sur le nombre de quarts de cube effectués soit le plus près possible du rapport du nombre total de quarts de choc sur le nombre total de quarts de cube durant cette période. Si on considère une période de planification de quatre semaines, le nombre total de quarts de choc est de 56 et le nombre total de quarts de cube est de 104. Ainsi, la valeur souhaitée est 56/104 ou 7/13. Soit n_R le nombre de quarts de cube effectués et n_T le nombre de quarts de choc effectués. L'idéal est donné par l'égalité suivante :

$$\frac{n_T}{n_R} = \frac{7}{13},$$

ou encore

$$13n_T - 7n_R = 0.$$

La valeur de la pénalité, notée P_{RT}, associée à cette contrainte est donnée par:

$$P_{RT} = \begin{cases} (13n_T - 7n_R)W^+_{TR}, & \text{si } 13n_T - 7n_R \geq 0, \\ -(13n_T - 7n_R)W^-_{TR}, & \text{sinon,} \end{cases}$$

où W^-_{TR} est le poids associé dans le cas où la quantité $(13n_T - 7n_R)$ est négative et W^+_{TR} est le poids associé dans le cas contraire.

- **Le rapport jour/soir.** Pour les mêmes raisons que précédemment, sur la même période de planification, il est souhaitable d'avoir le rapport du nombre de quarts de soir sur le nombre de quarts de jour le plus près possible de 84/88 ou 21/22. Soit n_D le nombre de quarts de jour effectués et n_E le nombre de quarts de soir effectués. L'idéal est donné par l'égalité suivante :

$$\frac{n_E}{n_D} = \frac{21}{22},$$

ou encore

$$22n_E - 21n_D = 0.$$

La valeur de la pénalité, notée P_{DE}, associée à cette contrainte est donnée

par:

$$P_{DE} = \begin{cases} (22n_E - 21n_D)W_{DE}^{+} & \text{si } 22n_E - 21n_D \geq 0, \\ -(22n_E - 21n_D)W_{DE}^{-} & \text{sinon,} \end{cases}$$

avec W_{DE}^{-} est le poids associé dans le cas où la quantité $(22n_E - 21n_D)$ est négative et W_{DE}^{+} est le poids associé dans le cas contraire.

Pour satisfaire au mieux les contraintes de buts du problème, celles-ci sont intégrées dans la fonction objectif. Chacune d'elles est pondérée selon son importance par rapport aux autres. Nous avons ainsi une fonction à plusieurs objectifs. Pour contourner le caractère conflictuel du problème posé et faciliter la mise en oeuvre, nous avons opté pour la méthode d'agrégation des objectifs, transformant ainsi le problème à plusieurs objectifs en un problème à un seul objectif qui peut être résolu par la méthode du simplexe. Le meilleur horaire est celui qui minimise cette fonction de pénalité globale. L'expression de la fonction économique est alors de la forme :

$$Minimiser \sum_{i \in I} \sum_{h \in H_i} c_i^h x_i^h$$

où

$$c_i^h = P_{RT} + P_{DE}.$$

La valeur de l'expression de c_i^h représente la somme des déviations par rapport à chacune des contraintes de buts associées à l'horaire h.

Le modèle du problème maître est alors :

$$Minimiser \sum_{i \in I} \sum_{h \in H_i} c_i^h x_i^h$$

sous les contraintes:

$$\sum_{i \in I} \sum_{h \in H_i} \delta_h^k x_i^h = 1, \quad \forall\, k \in K, \tag{4.1}$$

$$\sum_{h \in H_i} x_i^h = 1, \quad \forall\, i \in I, \tag{4.2}$$

$$x_i^h \in \{0, 1\} \tag{4.3}$$

4.2 Coût réduit associé

Le fonctionnement de la technique de génération de colonnes consiste à produire des colonnes potentiellement candidates pour le problème maître. L'expression du coût réduit qui constitue une base de communication entre le problème maître et le problème auxiliaire est la suivante:

$$\overline{c}_i^h = c_i^h - \textstyle\sum_{k \in K} \delta_h^k \pi_k - \mu_i, \quad \forall\, i \in I, \forall\, h \in H_i,$$

où $(\pi_k)_{k \in K}$ relatif à (4.1) et $(\mu_i)_{i \in I}$ relatif à (4.2) sont les vecteurs correspondant aux valeurs duales du problème maître relaxé.

4.3 Problème auxiliaire

Dans cette section, nous allons présenter le modèle basé sur la programmation par contraintes (PC) utilisé pour générer les colonnes du problème maître. Le problème consiste à générer des horaires individuels pour chaque médecin en tenant compte des spécificités de chacun. Chaque horaire généré devra être intéressant

au sens des coûts réduits pour aller, si possible, vers des solutions de meilleure qualité. Pour ce faire, nous pouvons générer des horaires sans tenir compte des coûts réduits et ensuite sélectionner ceux qui sont intéressants pour les ajouter au problème maître. Cette méthode présente l'inconvénient d'être très lente et non sûre. Nous avons alors défini une contrainte supplémentaire pour que tous les horaires soient intéressants au sens des coûts réduits. Dans notre cas, il sera question de générer, pour chaque médecin, des horaires respectant les contraintes liées au travail et la contrainte de coût réduit négatif, étant donné que la fonction de coût est à minimiser.

Le nombre de colonnes à ajouter au problème maître, pour un médecin donné, est quelconque. Nous pouvons ajouter l'ensemble des horaires intéressants générés ou n'en ajouter qu'une partie.

Le problème peut se formuler de la manière suivante : étant donné un médecin parmi le personnel, une période de planification, un ensemble de quarts de travail et de repos, ainsi qu'un ensemble de contraintes à respecter (règles générales, règles individuelles et coût réduit négatif), affecter au médecin en question un quart possible pour chaque jour de la période de planification. Nous présentons les variables et les contraintes du modèle, puis les stratégies de recherche qui permettront d'orienter l'engin de résolution vers l'obtention efficace de solutions.

4.3.1 Variables

Dans cette section, nous présentons les variables de décision, ainsi que les variables liées aux coûts réduits, que nous avons utilisées pour modéliser notre problème.

Variables de décision

Dans notre cas, les horaires des médecins sont traités indépendamment les uns des autres. Nous travaillerons seulement sur les variables d'un horaire individuel que l'on notera X. Pour modéliser le problème, nous associons à chaque jour j de la période de planification, une variable contrainte x_j, qui a comme domaine les différentes valeurs de quarts. Donc, pour tout jour $j \in P$ (période de planification), nous définissons une variable de décision x_j qui prend ses valeurs dans l'ensemble des quarts réalisables pour indiquer quel est le quart travaillé par le médecin le jour j. Si n est le nombre de jours de la période de planification, le vecteur de variables de décision sera alors $X = (x_1, x_2, \ldots, x_n)$.

Variables liées aux coûts réduits

Ces variables sont introduites pour garantir que chaque horaire généré est candidat potentiel pour le problème maître. La relation entre ces variables et les variables de décision est donnée par :

$$\overline{c} = c(X) - \sum_{k \in K} \delta_X^k \pi_k - \mu,$$

où $c(X)$ est le coût de l'horaire lié aux variables de décision X, δ_X^k est la variable binaire liée au vecteur de décision X, π_k la valeur de la variable duale relative

à (4.1) et enfin, μ est la valeur de la variable duale liée au médecin concerné, correspondant à (4.2).

4.3.2 Contraintes

Dans cette section, nous présentons les contraintes dures considérées pour notre planification et celles liées aux coûts réduits.

Pré-affectations

Un médecin peut se voir attribuer à priori des quarts de travail sur la période de planification. Ces quarts sont systématiquement à respecter. Notre solution doit garantir le respect de toutes les pré-affectations. Si A_F est l'ensemble des pré-affectations ou l'ensemble des affectations fixées, notre contrainte s'exprimera comme suit:

$$x_j = q, \quad \forall (j, q) \in A_F,$$

où (j, q) représente une pré-affectation du quart q au médecin le jour j.

Affectations interdites

Cette contrainte concerne les quarts qu'il ne faut jamais attribuer pour certains jours de la période de planification à un médecin donné. Notre solution devra respecter toutes les affectation interdites. Soit A_I l'ensemble des affectations interdites alors notre contrainte s'exprimera comme suit:

$$x_j \neq q, \quad \forall (j, q) \in A_I.$$

Quarts réalisables

Les quarts attribuables à un médecin donné sont fonction du niveau de qualification de celui-ci et de son expérience. Soit Q_R l'ensemble des quarts attribuables au médecin, l'expression de notre contrainte sera alors :

$$x_j \neq q \quad \forall q \notin Q_R, \quad \forall j \in P$$

Quarts du lendemain

Pour un enchaînement de quarts efficace et réalisable, certaines règles sont définies entre les quarts de deux jours successifs. Si un quart est affecté un jour donné, il ne faut affecter que les quarts réalisables le jour suivant. Soit Q_k l'ensemble des quarts possibles le jour suivant un quart k et soit n le nombre de jours de planification, alors notre contrainte sera exprimée comme suit :

$$x_j = k \Rightarrow x_{j+1} \neq q, \quad \forall q \notin Q_k, \quad \forall j \in \{1, 2, ..., n-1\}.$$

Quarts consécutifs

Un médecin ne doit pas assurer consécutivement, un nombre quelconque de fois, des quarts de même type. Pour ce faire, une borne minimale et une borne maximale par type de quarts sont définies pour chaque médecin. Soient $\{T_1, T_2, ..., T_m\}$, l'ensemble des types de quarts considérés ($T_i \cap T_j = \emptyset, \quad i \neq j$), S_{min} et S_{max} les deux vecteurs de tailles m contenant, respectivement, les valeurs des bornes

inférieures et les valeurs des bornes supérieures relativement à chaque type de quart. Notre solution doit assurer que le nombre d'affectations successives de quarts de même type soit entre les bornes définies pour ce type. L'expression de cette contrainte est alors donnée par :

stretch(X, S_{min}, S_{max}).

Charge de travail

Les charges de travail sont relatives à un médecin donné et sont exprimées en heures. Pour chaque médecin est définie une liste de contraintes de charge de travail. Chaque contrainte est exprimée par une période $J \subset P$, une charge minimale $minH$ et une charge maximale $maxH$. Étant donné que le nombre de durées de quarts différentes est inférieur au nombre total de quarts possibles, nous utilisons alors les variables de type en regroupant dans chaque type les quarts de même durée. Dans notre contexte, les quarts de travail durent huit ou quatre heures (0 heure est associé pour un jour de repos). Soient

- d un tableau de dimension égale au nombre total de quarts (y compris le repos), tel que $d[q]$ représente la durée en heures du quart q;
- Y un vecteur de variables de durées, qui transforme chaque valeur de quart de la variable de décision x_j en durée correspondante tel que $y_j = d[x_j]$;
- $M = (m_0, m_4, m_8)$ un vecteur des variables de multiplicité des durées dans Y.

L'expression de la contrainte sera alors :

$Y = d(X) \wedge \mathbf{distribute}(M, Y) \wedge \mathbf{ext}(M, \Gamma)$

où l'ensemble des couples admissibles Γ est donné par:

$\Gamma = \{(m_8, m_4) | \in [0..\overline{m_8}] \times [0..\overline{m_4}] \quad | \quad minH \leq 8m_8 + 4m_4 \leq maxH\}$

où $\overline{m_8} = \lfloor maxH/8 \rfloor$ et $\overline{m_4} = \lfloor maxH/4 \rfloor$.

Exemple

On suppose qu'au cours d'une période $J \subseteq P$, la charge de travail d'un médecin donné doit être comprise entre 20 et 40 heures. Pour évaluer la charge horaire, nous construisons le vecteur de durées Y en associant à chaque valeur de x_j, représentant un quart donné, la durée y_j correspondante. Nous obtenons ainsi un vecteur de variables de durées $Y = d(X)$. Étant donné que la multiplicité de la durée 4 ne peut pas être supérieure à $\lfloor 40/4 \rfloor$ et celle de durée 8 ne peut pas excéder $\lfloor 40/8 \rfloor$ et qu'il n'y ait pas de restriction sur la multiplicité de la durée 0 (repos), on applique alors la contrainte globale $distribute(M, Y)$ pour que ces conditions soient respectées. L'ensemble des solutions admissibles Γ est donnée par l'ensemble des couples (m_4, m_8) vérifiant la double inégalité :

$20 \leq 4m_4 + 8m_8 \leq 40$, avec $m_0 + m_4 + m_8 = |J|$, soit

$\Gamma = \{(7,0), (6,0), (6,1), (5,0), (5,1), (5,2), (4,1), (4,2), (4,3), (3,1),$

$(3,2), (3,3), (2,2), (2,3), (2,4), (1,2), (1,3), (1,4), (0,3), (0,4), (0,5)\}$ en considèrant une semaine de travail. À noter que la multiplicité de repos est donnée par le nombre de jours de la période considérée, c'est-à-dire 7, moins la somme des

deux valeurs du couple.

Coûts réduits

Pour que l'horaire généré soit une colonne candidate potentielle, il faut contraindre la variable relative aux coûts réduits à être négative d'où la contrainte suivante:

$$\overline{c} \leq -\overline{c}_{max}$$

i.e

$c(X) - \sum_{k \in K} \delta^k(X)\pi_k - \mu \leq -\overline{c}_{max}$. avec $\overline{c}_{max}$ une valeur positive que l'on peut choisir très proche de zéro.

Pour l'écriture de cette contrainte, il suffit d'exprimer la variable correspondant au coût de l'horaire, $c(X)$, qui est liée directement aux variables de décisions X. Pour ce faire, nous introduisons certaines variables supplémentaires. Soit Y^t un vecteur de variables qui associe à chaque variable de décision x_j une valeur 1 si le type de sa valeur est t et 0 sinon.

$$y_j{}^t = \begin{cases} 1, & \text{si } x_j \text{ est de type } t, \\ 0, & \text{sinon,} \end{cases}$$

où :

$t \in \{cube, choc, jour, soir\}$

sum $< Y^t, S^t >$ fait alors correspondre S^t au nombre de quarts de type t dans X.

La variable liée au coût de l'horaire, $c(X)$, correspond au cumul des déviations positives pondérées, par rapport aux buts fixés (voir section 4.1.3).
En définissant $L_{DE} \equiv (22S^{\text{soir}} - 21S^{\text{jour}} \geq 0)$ et $L_{TR} \equiv (13S^{\text{choc}} - 7S^{\text{cube}} \geq 0)$, nous avons alors l'équivalence suivante :

$$c(X) \equiv ((L_{TR}W^{+}_{TR}) + (\overline{L}_{TR}W^{-}_{TR})) + ((L_{DE}W^{+}_{DE}) + (\overline{L}_{DE}W^{-}_{TR})).$$

L'expression de la contrainte des coûts réduits est ainsi complétée.

4.3.3 Stratégies de recherche

Dans cette section, nous présentons les différentes stratégies de recherche utilisées.

Stratégie 1 basée sur les valeurs duales

Cette stratégie est basée sur les valeurs duales et l'aléatoire. Elle vise à avoir des horaires individuels avec un coût réduit le plus négatif possible. Ainsi, pour chacun des jours de la période de planification, on essaie de choisir d'abord le quart correspondant à la plus petite valeur de la variable duale associée aux contraintes de quotas (4.1). Selon l'expression du coût réduit (voir section 4.2), c'est en effet ce quart qui contribue le plus à rendre le coût réduit négatif. Afin de favoriser ce choix, pour chaque jour, nous ordonnons les quarts dans l'ordre croissant des valeurs duales en commençant par celui qui correspond à la plus petite valeur de la variable duale du jour en question. Nous associons alors la plus forte probabilité , qui est $p_0 = 0.4$ pour le quart correspondant à la plus petite valeur duale, $p_1 = 0.15$ pour la seconde, $p_2 = 0.10$ pour la troisième et enfin la même valeur 0.06 pour chacun

des quarts restants de l'ensemble des quarts Q (avec $\sum_{q \in Q} p_q = 1$).

Procédé Concernant cette stratégie, notre choix de variables consiste à choisir la première variable non encore instanciée. Soit x_j la variable à instancier, soit T un tableau de dimension égale au nombre de quarts possibles tel que $T_q = 100 \sum_{k=0}^{q} p_k$ et soit D le vecteur contenant les quarts ordonnés du plus intéressant au moins intéressant au sens des coûts réduits. Le procédé est alors défini comme suit (par convention, on pose $T_{-1} = 0$):

Répéter

-générer un nombre l aléatoire entre 0 et 99;

-le quart choisi $q = D_k$ *est celui qui correspond à*

$T_{k-1} \leq l < T_k$

Jusqu'à ce que $q \in D(x_j)$*;*

Retourner q;

Stratégie 2 basée sur la coopération entre le problème maître et le sous problème

Pour une meilleure coopération entre le problème maitre et le sous problème, nous avons proposé une stratégie visant à essayer de satisfaire la contrainte rigide de quotas du problème maître. Cette dernière consiste à affecter, pour chaque jour de la période de planification, une et une seule fois chacun des quarts de travail. En d'autres termes, un quart affecté à un médecin donné ne devrait pas être affecté à un autre médecin et ce pour chacun des jours concernés. Pour ce faire, il est

important de commencer par la variable ayant le minimum de valeurs dans son domaine. Pour instancier une variable, on essaie d'abord une des valeurs de quarts non encore choisie.

Procédé

Soit R un tableau de variables de dimension égale au nombre de jours de la période de planification tel que $D(r_j)$ contient les différents quarts à assurer pour le jour j. Soit x_j la variable à instancier correspondant au jour j. Le procédé est alors :

Étape 1: On essaie de choisir une valeur q telle que

$q = Min_{v_j \in D(r_j) \cap D(x_j)} v_j$

si succès, enlever la valeur q du domaine de r_j

Étape 2: Si échec à l'étape 1, on choisit q telle que

$q = Min_{v_j \in D(x_j)} v_j$

Retourner q;

CHAPITRE 5

IMPLÉMENTATION ET RÉSULTATS

Dans ce chapitre, nous donnons une brève présentation du langage utilisé, l'implémentation de la solution proposée, ainsi que quelques résultats obtenus.

5.1 Présentation du langage

Notre choix s'est porté sur *Ilog Concert Technology Concert* qui permet d'intégrer la programmation mathématique et la programmation par contraintes. Cet environnement de développement est parfaitement adéquat pour notre approche hybride qui combine ces deux techniques.

Nous avons réalisé notre projet avec la version 1.1 de *Concert* [24]. Celle-ci combine les versions 7.1 de *Cplex* [26] et 5.1 de *Solver* [25].

La technologie utilisée offre des classes C++, aussi bien pour la représentation de modèles que pour l'implémentation des algorithmes de résolution. Elle offre des moyens appropriés pour exploiter la technique de génération de colonnes, en particulier pour traiter le problème maître. En effet, comme nous le verrons plus loin, les expressions offertes nous ont permis une prise en charge aisée.

Une application *Concert* passe par la création d'un environnement,
la construction d'un ou plusieurs modèles, l'extraction de modèles, la résolution

du problème pour enfin finir avec l'accès aux résultats. Nous commençons par créer un environnement (instance de la classe `IloEnv`). Chaque modèle (instance de la classe `IloModel`) et chaque algorithme de résolution (instance d'une sous classe de `IloAlgorithm`, `IloSolver` ou `IloCplex`) défini doit appartenir à cet environnement.

Pour résoudre un *problème d'optimisation*, nous commençons par construire le modèle (instance de la classe `IloModel`) avec ses variables de décision (instances de la classe `IloNumVar`), les contraintes que ces dernières doivent satisfaire (instances de la classe `IloRange` ou instance de `IloConstraint`), ainsi qu'une fonction objectif à optimiser (instance de la classe `IloObjective`). L'étape suivante consiste à appliquer un algorithme (Instance de la classe `IloAlgorithm`) pour résoudre le modèle.

5.2 Implémentation du problème maître

Dans cette section, nous présentons comment nous avons exprimé le problème maître en *Concert*.

5.2.1 Variables et contraintes

Étant donné qu'il y a lieu de générer des horaires pour tous les médecins et que l'on ne connaît pas, à priori, le nombre d'horaires à générer pour chacun d'eux, nous avons défini un tableau de dimension égale au nombre de médecins (nbreMed)

où chaque composante est un tableau à dimension non fixée (*Concert* permet de déclarer un tableau extensible avec une dimension quelconque au départ, dans notre cas 0). Voici alors la déclaration des variables :

```
IloNumVarArray2 Medecin(env,nbreMed);
for (int m=0; m<nbreMed;m++)
Medecin[m]=IloNumVarArray(env);
```

`env` est l'identificateur de l'instance de IloEnv représentant l'environnement de l'application.

La contrainte qui assure que chaque quart du jour soit assuré une et une seule fois est donnée par:

```
IloRangeArray Quarts = IloAdd(horOpt,IloRangeArray(env,Inf,Sup));
```

`horOpt` est l'identificateur de l'instance de IloModel représentant le modèle du problème maître.

Cette déclaration veut dire que `Quarts` est le tableau de contraintes tel que, pour toute ligne `k` relative à un quart donné, nous avons `Inf[k]<= Quarts[k]<=Sup[k]`. `Inf[k]` et `Sup[k]` représentent respectivement le quota minimal et le quota maximal pour le quart `k`. Dans notre cas `Inf[k]=Sup[k]=1`. Cette instruction ajoute la partie (4.1) des contraintes au modèle maître `horOpt`.

Enfin, la contrainte qui assure que chaque médecin aura un et un seul horaire est exprimée comme suit:

```
IloRangeArray ContHorUniq =IloAdd(horOpt,
```

```
                    IloRangeArray(env,nbreMed,1,1));
```

Il y a une contrainte par médecin, ce qui explique la dimension `nbreMed`. Toutes les lignes sont bornées inférieurement et supérieurement par 1. Cette instruction ajoute la partie (4.2) des contraintes au modèle maître `horOpt`.

5.2.2 Ajout de colonne

Une colonne est définie par le coefficient de la fonction objectif, les coefficients binaires de l'horaire généré (partie 4.1 des contraintes) et la spécification du médecin concerné (partie 4.2 des contraintes). L'ajout d'une colonne se fait alors par :

```
Medecin[i].add(IloNumVar(horObj(valCout(med[i],i, mData)) +
                    Quarts(newH)+ContHorUniq[i](1),0,1,ILOFLOAT));
```

`Medecin[i].add(...)` signifie que la variable est à ajouter aux colonnes déjà générées pour le médecin `i`. `horObj(valCout(med[i],i, mData))` correspond à l'affectation de la valeur `valCout(med[i],i, mData)` au coefficient de la variable en question de la fonction objectif `horObj`. `Quarts` représente la partie contrainte liée aux quarts. `Quarts(newH)` ajoute la partie de la colonne de coefficient correspondant au nouvel horaire `newH`. `ContHorUniq[i](1)` permet de remplir la deuxième partie des contraintes en mettant 1 à la position `i` (le reste est mis automatiquement à 0). `0,1` pour indiquer que la variable est binaire. `ILOFLOAT` précise le type de la nouvelle variable comme étant réelle.

5.3 Fonctionnement du système

Dans cette section, nous présentons comment fonctionne en général notre système et les différentes phases qui le composent.

La recherche de la solution passe par trois phases essentielles. **La phase d'initialisation** consiste à trouver une ou plusieurs solutions initiales en générant un ensemble d'horaires par médecin, de telle sorte qu'il y ait au moins une solution vérifiant que chaque médecin a un seul horaire et que chaque quart de chaque jour de l'horizon de planification est assuré une et une seule fois. Nous avons la possibilité de démarrer le système avec une ou plusieurs solutions fournies. Nous avons aussi la possibilité de laisser le système générer une ou plusieurs solutions initiales. Pour ce faire, une contrainte de quotas est utilisée uniquement dans la phase d'initialisation. Cette façon de faire, permet d'initialiser le processus itératif d'optimisation avec une solution respectant les contraintes imposées à chacun des médecins et par conséquent proche de la réalité. En somme, durant cette phase, nous assurons un point de départ pour le problème maître et aussi nous appliquons toutes les contraintes dures que chacun des horaires individuels doit respecter au niveau du sous-problème. **La phase d'optimisation** est ensuite appliquée pour améliorer, si possible, la solution issue de la phase d'initialisation. Le processus d'amélioration de la solution, est répété aussi longtemps qu'il y a des colonnes (ou des horaires) qui respectent, en plus des contraintes spécifiques, la condition de coûts réduits négatifs. Nous avons la possibilité d'arrêter le processus après un

certain nombre d'itérations fixé à l'avance. Concernant les contraintes du sous-problème, celles-ci sont ajoutées avant d'entamer la phase d'optimisation. La contrainte de coûts réduits est ajoutée à chaque début d'une itération d'optimisation et retirée à la fin de cette dernière. Une itération au niveau de cette phase peut être présentée comme suit:

- prise en compte des valeurs duales
- ajout de la contrainte de coûts réduits au sous-problème;
- génération de colonnes et augmentation du problème maître;
- résolution du problème maître augmenté;
- suppression de la contrainte de coûts réduits.

La solution obtenue à la fin de cette phase est la solution réelle du problème maître relaxé.

Notre démarche consiste à résoudre le problème linéaire maître relaxé à l'optimalité pour ensuite, procéder au *"Branch and Bound"* pour chercher la solution entière pour notre problème. **La phase d'intégralité** prend ainsi en charge la recherche d'une solution entière. De plus, Cette dernière phase peut être exploitée pour chercher d'autres solutions, si possible, de qualité égale ou inférieure à la meilleure solution entière obtenue.

5.4 Description des données

Nous reprenons le contexte décrit dans [6] en nous intéressant particulièrement aux données du premier mois, soit durant la période du 5 janvier au 2 février 1998.

Il y a beaucoup de vacances ; donc, la méthode cherchera la solution dans un contexte de personnel réduit. De plus, certains médecins ont des disponibilités très limitées . Le tableau 5.1 présente les disponibilités et les requêtes de chacun des médecins.

Comme décrit au chapitre 4, nous avons considéré les rapports cube/choc et jour/soir pour définir notre fonction objectif. Une valeur souhaitée est définie pour chacun des rapports. Nous avons repris les mêmes poids que ceux utilisés dans [6] pour mesurer la qualité d'un horaire. Les deux objectifs sont considérés avec la même importance. Un poids de valeur 4 est utilisé aussi bien dans le cas de l'excès ou de déficit par rapport aux valeurs souhaitées.

Tableau 5.1: Disponibilités et requêtes des médecins

Médecin 1: Junior consécutivité de nuits possible vacances: 5-11 janvier	Médecin 2: Junior consécutivité de nuits possible vacances: 10-18 janvier
Médecin 3: Junior nuits séparées pas de mercredis	Médecin 4: Junior nuits collées vacances: 17 janvier-1er février
Médecin 5: Junior nuits collées	Médecin 6: Sénior nuits séparées
Médecin 7: Sénior nuits séparées pas de lundis	Médecin 8: Sénior nuits collées vacances: 5-11 janvier
Médecin 9: Sénior nuits séparées	Médecin 10: Sénior consécutivité de nuits possible pas de mercredis vacances: 17-25 janvier
Médecin 11: Sénior nuits collées pas de mardis	Médecin 12: Sénior nuits séparées vacances: 17-25 janvier
Médecin 13: Sénior nuits collées	Médecin 14: Sénior nuits séparées
Médecin 15: Sénior pas de nuits ni de soirs aucun quart du lundi	Médecin 16: Sénior pas de nuits pas de mardis vacances: 10-18 janvier
Médecin 17: À temps partiel affecté selon ses disponibilités	Médecin 18: À tepms partiel affecté selon ses disponibilités

5.5 Quelques résultats illustratifs

Le test est fait sur une période de quatorze jours. Toutes les contraintes sont considérées à l'exception de la contrainte de charge de travail. Pour illustrer notre approche hybride, nous présentons l'impact de chacune des stratégies utilisées au niveau du problème maître et du sous-problème. D'une part, nous nous intéressons à l'évolution de la fonction objectif et à l'intégralité de la solution et d'autre part, à la résolution du sous-problème en termes de rapidité d'exécution et du nombre d'échecs. La figure 5.1 permet de comparer les différentes stratégies concernant l'évolution de la fonction objectif. La première, notée "Defaut", concerne la recherche par défaut de Solver, la stratégie "Dual" est celle qui est basée sur les valeurs duales et enfin la stratégie "Maitre" est celle qui est basée sur la coopération entre le problème maître et le sous-problème (voir le chapitre 4). Le tableau 5.2 donne pour chacune des stratégies utilisées.

- les valeurs de la fonction objectif correspondant à la solution initiale("Init."), la solution réelle ("Reel.") et à la solution entière ("Ent.");
- la moyenne des temps d'exécution ("T-M") et le nombre moyen d'échecs ("Ech-M") obtenus dans la résolution du sous-problème.

Tableau 5.2: Tableau comparatif

Strat.	*Init.*	*Reel.*	*Ent.*	$T-M$	$Ech-M$
Defaut	28816	28816	28816	3.68	200.41
Dual	28816	12028.4	28816	1.84	0.12
Maitre	28816	13481	13952	2.28	37.27

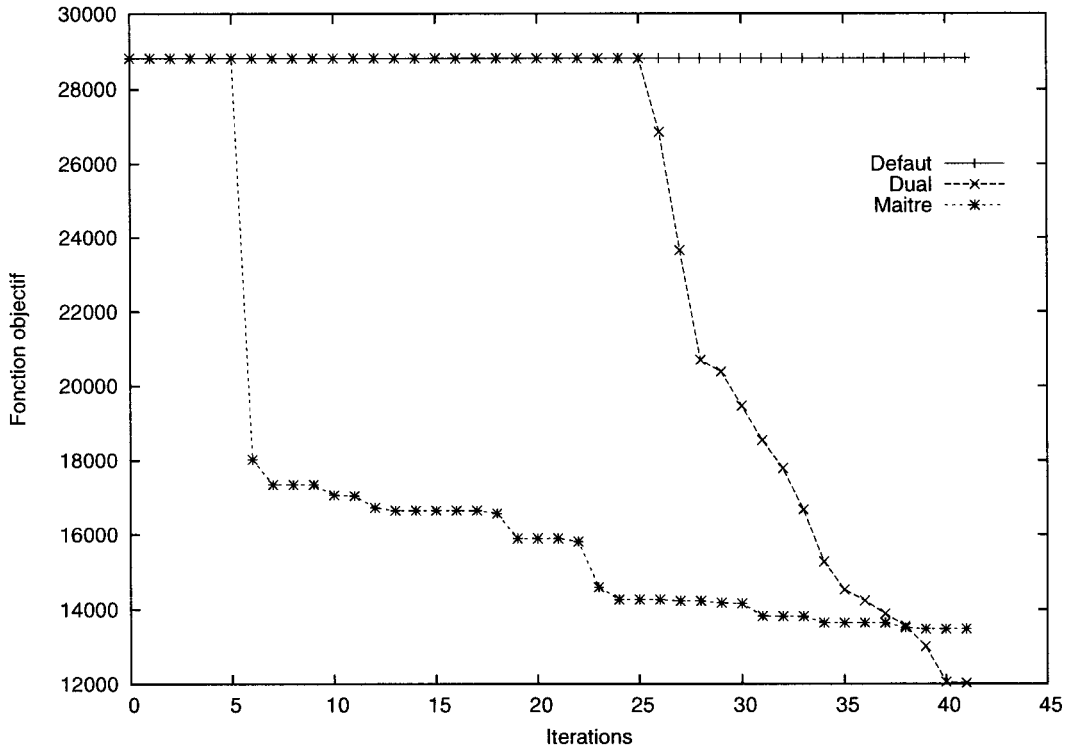

Figure 5.1: Comparaison

Analyse des résultats

Un certain nombre de conclusions se dégagent des résultats:

- Stratégie "Defaut": On constate une stagnation de la fonction objectif.

  De plus, la résolution est relativement lente avec un grand nombre d'échecs.

Ce qui peut s'expliquer par le fait que la stratégie par défaut qui consiste à choisir d'abord la première variable non encore instanciée en essayant les valeurs de la plus petite à la plus grande, est loin d'etre bien adaptée à la nature de notre problème.

- Stratégie "Dual": La fonction objectif stagne au départ pour ensuite décroître d'une manière significative. Les colonnes générées aux premières itérations rentrent dans la base du PL sans faire partie de la solution de ce dernier. Quand elles commencent à faire partie de la solution, la valeur de la fonction objectif diminue considérablement étant donné que celles-ci correspondent au coût réduit le plus négatif possible. Etant donné la stratégie adoptée, la contrainte de coûts réduits, qui est assez rigide, est souvent vérifiée ce qui peut expliquer en partie cette performance en temps et en échecs par rapport aux autres stratégies. Cependant cette manière de faire ne favorise pas l'obtention de solutions entiéres.

- Stratégie "Maitre": Étant donné que cette stratégie vise à satisfaire les contraintes du problème maître, les colonnes générées ont une forte chance de faire partie de la solution. Ceci explique la diminution de la fonction coût aux premiéres itérations d'une part et l'obtention de solutions entières d'autre part. Le temps moyen et le nombre d'échecs sont relativement satisfaisants.

CONCLUSION

La confection d'horaire est un problème auquel se sont intéresses plusieurs organisations. Les méthodes appliquées sont diverses. Notre approche a combiné les forces d'expression de la programmation par contraintes et la généricité de la programmation mathématique. La téchnique de génération de colonnes constitue le support d'application de notre approche hybride. Le problème maître est pris en charge par la programmation mathématique, alors que le sous-problème est résolu par la programmation par contraintes.

La résolution du modèle mathématique proposé présente deux difficultés principales qui s'ajoutent au caractère combinatoire du problème à l'étude:
la dégénérescence et l'intégralité. Ainsi, après un temps d'exécution significatif dans la phase d'optimisation, la valeur de la fonction objectif demeure inchangée. De plus, la recherche d'une solution entière échoue, fournissant souvent comme meilleure solution entière la solution initiale. La constatation générale est la suivante: le problème maître ainsi formulé est trop rigide. Les contraintes (4.1) constituent la partie difficile à satisfaire. Pour contrer ces problèmes, nous avons proposé des stratégies de recherche de solutions qui permettent une meilleure coopération entre le problème maître et le sous problème.

Nous avons proposé d'abord une stratégie basée sur les valeurs duales. Celle-ci vise essentiellement à faire évoluer le coût. En effet, dans la majorité des cas,

le test est concluant puisque nous avons observé dans les résultats présentés une amélioration significative de la fonction objectif. À noter que cette stratégie est générale et peut être exploitée dans les cas où l'optimisation est capitale et le problème d'intégralité est secondaire. Même si cette stratégie permet de faire évoluer la fonction objectif dans la relaxation continue, le problème d'identifier de meilleures solutions entières demeure. La deuxième stratégie vise à aider le problème maître à trouver des solutions entières. Pour ce faire, nous avons orienté la recherche pour essayer de trouver des horaires tels que les contraintes du problème maître soient vérifiées et qu'en même temps la fonction objectif évolue. Nous acceptons alors d'augmenter le problème maître non seulement avec des colonnes intéressantes au sens des coûts réduits mais aussi avec des colonnes permettant de favoriser l'obtention de solutions entières. Ces dernières servent à faire face au problème d'intégralité. En effet, le constat est positif. Comme le montrent les résultats, avec cette dernière stratégie, la fonction objectif évolue et la solution entière obtenue est souvent différente de la solution initiale.

Ceci dit, il reste à mettre en épreuve toutes ces idées pour évaluer de plus près l'apport de chacune d'elles. Par ailleurs, nous n'avons pas implémenté toutes les contraintes pour que le système réalisé soit opérationnel pour l'Hôpital Sacré-Coeur de Montréal. Enfin, nous espérons avoir fait une base suffisante pour des travaux futurs qui viendront compléter le système.

BIBLIOGRAPHIE

[1] Anzai M., Miura Y., *"Computer Program for Quick Work Scheduling of Nursing Staff"*, Medical Information, Vol. 12, pp. 43-52 (1987).

[2] Bartholdi J. J., Orlin J. B., Ratliff H. D., *"Cyclic Scheduling Via Integer Programs With Circular Ones"*, Operations Research, Vol. 28, pp. 1074-1085 (1980).

[3] Barnhart C., Johnson E. L., Nemhauser G. L., Savelsberg M. W. P., Vance P. H. *"Branch-and-Price: Column Generation for Solving Huge Integer Programs"*, Operations Research, 46(3), pp. 316-329. (1998).

[4] Barták R., *"Constraint programming: In Pursuit of the Holy Grail"*, Proc. of the Week of Doctoral Students (WDS99), MatFyzPress, Prague, 555-564, Juin 1999. (1999).

[5] Beaulieu H., Ferland J. A., Gendron B., Michelon P., *"A Mathematical Programming Approach for Scheduling Physicians in the Emergency Room"*, Health Care Management Science, Vol. 3, pp. 193-200 (2000).

[6] Beaulieu H., *"Planification de l'horaire des médecins dans une salle d'urgence"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (1998).

[7] Berrada I., *"Planification d'horaires du personnel infirmier dans un établissement hospitalier"*, Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, (1993).

[8] Bessière C., Régin J.-C., *"Refining the Basic Constraint Propagation Algorithm"*, Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'2001), pages 309-315, (2001).

[9] Bourdais S., *"Génération automatique d'horaires en milieu hospitalier"*, Mémoire de maîtrise, École Polytechnique de Montréal, (2003).

[10] Bourdais S., Galinier Ph., Pesant G., *"HIBISCUS: A Constraint Programming Application to Staff Scheduling in Health Care"*, Principles and Practice of Constraint Programming: Proceedings of the Ninth International Conference (CP'03), Kinsale (IE), Springer-Verlag Lecture Notes in Computer Science 2833, 153–167, (2003).

[11] Burns R. N., *"Manpower Scheduling With Variable Demands and Alternate Weekends Off"*, INFOR, Vol. 16, pp. 101-111 (1978).

[12] Burns R. N., Koop G. J., *"A Modular Approach to Optimal Multiple-Shift Manpower Scheduling"*, Operations Research, Vol. 35, No 1, pp. 100-110 (1987).

[13] Chan L. K., Falkenberg J., Rosenbloom E. S., *"Implementation Problems Of Nurse Preference Mathematical Programming Approach to Scheduling"*, Congressus Numerantium, Vol. 56, pp. 251-260 (1987).

[14] Dantzig G. B., Wolfe P., *"Decomposition Principle for linear programs"*, Operations Research 8, pp. 101-111, (1960).

[15] Desrosiers J., Dumas Y., Solomon M.M., Soumis F., *"Time Constrained Routing and Scheduling"*, Handbooks in Operations and Management Science 8, Volume on Network Routing, Elsevier, Amsterdam, pp. 35-139, (1995).

[16] Feuilletin W., *"Génération automatique d'horaires d'infirmières à l'aide de la programmation par contraintes"*, Publication CRT-2000-32, Centre de recherche sur les transports, Université de Montréal. (2000).

[17] Forget F., *"Confection automatisée des horaires de médecins dans une salle d'urgence"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (2002).

[18] Gagné E., *"Application d'une méthode exacte pour la génération d'horaires en soins infirmiers"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (1996).

[19] Gilmore P. C., Gomory R. E., *"A Linear Programming Approach to the Cutting Stock Problem"*, Operations Research 9, pp. 849-859. (1960).

[20] P. C. Gilmore, R. E. Gomory, *"A Linear Programming Approach to the Cutting Stock Problem: Part II"*, Operations Research 11, pp. 863-888. (1963).

[21] Glover F., Laguna M., *"Tabu Search"*, Kluwer, Boston (1997).

[22] Heus K., *"Gestion des plannings infirmiers, application des techniques de programmation par contraintes"*, Thèse de doctorat en informatique, Université Joseph Fourier - Grenoble, (1996).

[23] Ignizio J. P., *"Goal Programming and Extensions"*, Lexington Books, Massachusetts, (1973).

[24] ILOG S.A., *"Ilog Concert Technology 1.1"*, Paris France, (2001).

[25] ILOG S.A., *"Ilog Solver 5.1"*, Paris France, (2001).

[26] ILOG S.A., *"Ilog Cplex 7.1"*, Paris France, (2001).

[27] Jaumard B., Seme F. S., Vovor T., *"A Generalized Linear Programming Model for Nurse Scheduling"*, European Journal of Operational Research, Vol. 107, pp. 1-18, (1998).

[28] Labit P., *"IRIS : Amélioration d'une méthode de génération de colonnes pour la confection d'horaires d'infirmières"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (2000).

[29] Labbé S., Gendreau M., Lapierre S. D., Soriano P., *"A Tabu Search for Scheduling Physicians"*, Présenté à Informs, Atlanta, 3-6 Nov (1998).

[30] Lapierre S. D., Carter M. W., *"Scheduling Emergency Room Physicians"*, Publication CRT-99-35, Centre de recherche sur les transports, Université de Montréal, (1999).

[31] Marriott K., Stuckey P. J., *"Programming with Constraints, an Introduction"*, Cambridge, Massachusetts, MIT Press, (1998).

[32] Meyer auf' m Hofe, *"Solving Rostering Tasks by Generic Methods for Constraints Optimization"*, International Journal of Optimization of Foundations of Computer Science, Vol. 12, No 5, pp. 671-693 (2001).

[33] Miller H. E., Pierskalla W. P., Rath G. J., *"Nurse Scheduling Using Mathematical Programming"*, Operations Research, Vol. 24, No 5, pp. 857-870 (1976).

[34] Nabli I., *"Horaires du personnel infirmier générés avec approches heuristiques"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (1995).

[35] Ozkaraha I., Bailer J., *"Goal Programming Model Subsystem of a flexible Nurse Scheduling Support System"*, IIE Transactions, pp. 306-316 (1988).

[36] Pierskalla W. P., Brailer D. J., *"Applications of Operations Research in Health Care Delivery"*, Chapitre 13 de Handbooks in OR & MS, Vol. 6, édité par S. M. Pollock et al. (1994).

[37] Pesant G., *"A Filtering Algorithm for the Stretch Constraint"*, Principles and Practice of Constraint Programming (CP 2001), Springer-Verlag LNCS 2239, 183–195 (2001).

[38] Saadie M., *"Planification de l'horaire des médecins dans une salle d'urgence par la programmation par contraintes"*, Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, (2003).

[39] Sherali H. D., Soyster A. L., *"Preemptive and Non-Preemptive Multi-Objective Programming: Relationships and Counter Examples"*, Journal of Optimization Theory and Applications, Vol. 39, No. 2, pp. 173-186 (1981).

[40] Sherali H. D., *"Equivalent Weights for Lexicographic Multi-Objective Programs : Characterisation and Computation"*, European Journal of Operational Research, Vol. 11, pp.367-379, (1982).

[41] Régin J.-C., *"Generalized Arc Consistency for Global Cardinality Constraints"*, Proc. of American Association of Artificial intelligence (AAAI-96), 209–215, AAAI Press/MIT Press, (1996).

[42] Rousseau L. M., Pesant G., Gendreau M., *"A Hybrid Algorithm to Solve a physician Rostering Problem"*, In Second Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems, Paderborn, Germany, (2000).

[43] Smith L., Wiggins A., *"A Computer Based Nurse Scheduling System"*, Computers and Operations Research, Vol. 4, pp. 195-212 (1977).

[44] Steuer R. E., *"Multiple criteria optimization: Theory, Computation and Application "*, Wiley, (1986).

[45] Tibrewala R., Philippe D., Browne J., *"Optimal Scheduling of Two Consecutive Idle Periods"*, Management Science, Vol. 19, pp. 71-75 (1972).

[46] Trilling G., *"Génération automatique d'horaires de médecins de garde pour l'Hôpital Côte-des-Neiges de Montréal"*, Publication CRT-98-05, Centre de recherche sur les transports, Université de Montréal. (1998).

[47] Vassilacopoulos G. *"Allocating Doctors to Shifts in an Accident and Emergency Departement"*, Journal of the Operational Research Society, Vol. 36, pp.517-523. (1985).

[48] Warner D. M., *"Scheduling Nursing Personnel According to Nursing Preference : A Mathematical Programming Approach"*, Operations Research, Vol. 24, pp. 842-856 (1976).